International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-15 Issue-4, March 2026

Self-Healing Infrastructure: Autonomous LLM Agents for Real-Time Remediation of Configuration Drift and Security Misconfigurations in IaC Deployments

Harish Apuri, Madhan Mohan Reddy Chinthala, Shikher Goel, Mukesh Aurangabadkar, Charani Yepuri

Abstract: ***The application of Infrastructure as Code (IaC) has enhanced cloud environment scalability and automation, but configuration drift and security misconfigurations remain critical operational and security issues. Current drift detection and remediation solutions rely largely on reactive, rules-based, and human intervention; therefore, they are ineffective in dynamic, multi-cloud environments. This research aims to develop and deploy a self-healing infrastructure architecture that autonomously identifies and recovers from configuration drift and security misconfigurations in real time. The paper suggests the following to accomplish this: a new multi-agent architecture based on Large Language Models (LLMs), in which Drift detectors, security reasoners, root-cause analysers, remediation generators, and post-remediation validators operate within a closed-loop pipeline. To evaluate the framework, a publicly available IaC dataset (written in Terraform) of simulated drift situations is used. According to experimental results, the proposed LLM-agent system outperforms rule-based and semi-automated systems, with a drift detection rate of 96.8, a security misconfiguration detection rate of 95.2, and a mean time to remediation of 6.9 minutes. The framework is also very effective in reducing false positives and manual intervention, as well as getting high policy compliance. Such findings affirm the usefulness of autonomous LLM agents in empowering proactive, intelligent and scalable self-healing infrastructure management in contemporary cloud systems.***

Keywords: Self-Healing Infrastructure, Infrastructure as Code (IaC), Large Language Models (LLMs), Configuration Drift Remediation, Cloud Security Automation

Nomenclature:
IAM: Identity and Access Management
IR: Intermediate Representation
LLMs: Large Language Models
IaC: Infrastructure as Code
EDA: Exploratory Data Analysis
LLM: Large Language Model

Manuscript received on 01 February 2026 | First Revised Manuscript received on 27 February 2026 | Second Revised Manuscript received on 05 March 2026 | Manuscript Accepted on 15 March 2026 | Manuscript published on 30 March 2026.
*Correspondence Author(s)
Harish Apuri*, Department of IT, IT Induct Inc, Charlotte (NC), United States of America (USA). Email ID: hapuri1029@gmail.com, ORCID ID: 0009-0007-1534-9984
Madhan Mohan Reddy Chinthala, Department of IT, Franklin Info Tech, Charlotte (NC), United States of America (USA). Email ID: madhanreddychinthala@gmail.com
Shikher Goel, Department of IT, JPMorgan Chase, Jersey (New Jersey), United States of America (USA). Email ID: Shikher20goel@gmail.com
Mukesh Aurangabadkar, Department of IT, Spectrum, Denver (Colorado), Vanuatu. Email ID: Mukesh_a24@hotmail.com
Charani Yepuri, Independent Researcher, Department of IT, Hyderabad (Telangana), India. Email ID: Charaniy04@gmail.com

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open-access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

I. INTRODUCTION

The adoption of Infrastructure as Code (IaC) is fast becoming the standard for deploying, configuring, and operating modern clouds and multi-cloud environments. IaC frameworks like Terraform, AWS CloudFormation, and Ansible have become particularly useful for improving scalability, repeatability, and operational efficiency by enabling infrastructure to be described, deployed, and managed under version control using machine-readable configuration files. Nevertheless, as cloud environments grow in size and complexity, ensuring consistency between stated infrastructure conditions and actual runtime conditions is urgent. Configuration drift, where the actual state of infrastructure differs from the desired, code-defined state due to manual intervention, ad hoc fixes, automated updates, or external policy changes, is one of the most persistent problems in this context. Configuration drift is not just a convenient nuisance to the operation process, but it brings about severe security risks, regulatory breaches, and operational instability. Research has shown that a significant share of cloud security breaches stems from misconfigurations, including overly permissive access controls, unsecured storage, and unpatched services. Older drift detection systems-usually periodic scan-driven, or rule-based, or enforcement of fixed policies-are usually reactive, slow, and not applicable in real-time. Besides, these methods may reveal deviations, but they often depend on human intervention for alert decoding and remediation implementation, leading to delayed responses and high operational risk [1] [2] [3] [4] [5].

The idea of self-healing infrastructure has become one of the leading paradigms for addressing these challenges in recent years [6] [7] [8] [9] [10]. The self-healing systems are designed to automatically identify, diagnose, and fix errors or anomalies without human intervention, and are used to enhance resilience and minimise mean time to recovery (MTTR). The current self-healing solutions in cloud architecture are mostly based on predefined rules, threshold-based monitoring, or scarcity-automation scripts [11] [12] [13] [14] [15]. While useful with familiar failure patterns, they fail to reason about contexts, handle dynamic settings, and handle previously unknown misconfigurations, especially in heterogeneous IaC deployments across

© Copyright: All rights reserved.

multiple cloud providers and services.

With the development of Large Language Models (LLMs), intelligent automation in software engineering and cloud operations has gained new opportunities. LLMs have strong semantic understanding and contextual reasoning, as well as the ability to make decisions across a range of inputs, including unstructured data such as configuration files, logs, policy documents, and security advisories. Recent studies have examined the applications of LLMs in code generation, vulnerability analysis, DevOps support, and infrastructure optimisation. Nevertheless, the majority of current literature uses LLMs passively or in an advisory capacity, such as proposing fixes or creating remediation scripts, rather than implementing fully autonomous, closed-loop remediation systems.

The first significant weakness of modern LLM-based approaches is the absence of agent-based autonomy and real-time interoperability with IaC pipelines and runtime infrastructure. End-to-end lifecycle drift detection, impact analysis, remediation planning, execution, and validation cannot be treated using single-model prompting techniques. Moreover, available solutions rarely discuss security misconfigurations and configuration drift in tandem, as they are highly interdependent in practice with cloud environments. Another serious research limitation is the lack of intelligent system design to ensure constant monitoring of IaC-controlled infrastructure, to reason about deviations based on contextual knowledge, and to enable the real-time, autonomous execution of safe, verifiable remediation actions.

To fill this gap, the paper presents an example of a Self-Healing Infrastructure Framework, driven by Autonomous LLM Agents, that remediates configuration drift and security misconfigurations in real time during an IaC deployment. The suggested model uses a multi-agent design, with expert LLM agents working together to perform drift detection, security risk evaluation, root-cause analysis, remediation generation, and post-remediation validation. The system allows maintaining a steady balance between the intended and actual infrastructure state by combining runtime state monitoring with IaC specifications, policy constraints, and security best practices. In contrast to a rigid rule-based system, the suggested one will leverage the power of reasoning and the generality of LLMs to address complex, changing, and previously unseen misconfigurations.

A. Contributions and Novelty of the Work

The key contributions and novel aspects of this research are summarised as follows:

- Autonomous LLM-Agent Framework on Self-Healing IaC: This project proposed a new multi-agent LLM architecture that can detect and resolve configuration drift and security misconfigurations in an environment with IaC management, instead of taking an advisory or human-in-the-loop perspective.
- Real-Time Closed-Loop Remediation Pipeline: The proposed framework supports real-time monitoring, reasoning, remediation execution, and validation, unlike periodic scan-based systems, which ensure a consistent infrastructure at all times and compliant security.
- Coherent Handling of Drift and Security Misconfigurations. The research has offered a single model that addresses both configuration drift and security misconfigurations in combination and manages their interrelationships; the absence of such a model results in a fragmented remediation policy.
- Context-Aware and Policy-Compliant Remediation: The system can use IaC definitions to produce remediation actions that are safe, explainable, and policy-compliant by embedding IaC definitions with runtime states, organizational policies, and security best practices in the logic of the LLM.
- Generalizability to Multi-Cloud IaC Deployments: The framework has been formulated to be generically applicable to the heterogeneous IaC tools and cloud platforms, which are assumed to apply to real-life, large-scale cloud environments.

II. LITERATURE REVIEW

The review of LLM-guided phone GUI agents by G. Liu et al. (2025) [16] shows that these agents, available as scripts, have evolved into intelligent, adaptable systems. We introduce our background on major challenges, (i) low generality, (ii) heavy maintenance cost and (iii) a weak understanding of intentions and demonstrate the way in which such challenges are resolved in LLMs through multimodal perception, advanced language understanding and strong decision-making. We then suggest a taxonomy of basic agent structures (single-agent, multi-agent, plan-act), modelling styles (prompt engineering, training-based), and key datasets and benchmarks. Moreover, we describe task-related architectures, fine-tuning under supervision, and reinforcement learning plans that bridge the user's intent and GUI activity. Last but not least, there are open challenges (which we will discuss), such as dataset diversity, on-device deployment efficiency, user-oriented adaptation, and security, offering valuable insights into this fast-paced sphere. The paper outlines critical research gaps and provides a clear overview of the subject; therefore, it can be considered a reliable source for researchers and practitioners seeking to utilise LLMs to create scalable, convenient phone GUI agents. The repository of the papers examined during this survey will be hosted and updated regularly on the GitHub repository: https://github.com/PhoneLLM/Awesome-LLM-Powered-Phone-GUI-Agents.

Y. Liu et al. (2025) [17] propose the concept of zero-trust security for multi-LLM in EGI, representing a paradigmatic shift towards the principle of never trust, always verify. Our starting point is the systematic analysis of the security risks of the multi-LLM systems in the context of EGI. Then we present the vision for a zero-trust multi-LLM framework at EGI. The next step is to scan the main technical advances to enable zero-trust multi-LLM systems in EGI. In particular, we distinguish between the 0-trust security mechanisms at the model- and system-level. The former entails strong identification, context-based access control, and proactive maintenance, while the latter

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

entails blockchain-based management. Lastly, we determine key research directions. The survey is the first systematic study of zero-trust for multi-LLM systems, offering theoretical background and practical strategies. As suggested by Vyas and Mercangöz (2025 [18], the novel concept of industrial automation entails validating and reprompting architectures for large language model (LLM)-based autonomous control agents. The proposed system of agents comprises an operator, a validator, and a reprompter agent, enabling autonomous management of control tasks and adaptation to unexpected disturbances with zero human intervention. With validation and reprompting architectures, the framework enables agents to recover from errors and achieve perfect decision-making in real-time industrial environments. We assume that implementing this mechanism will improve performance and reliability across numerous LLM models, enabling fully autonomous systems to manage unforeseen challenges, which is a path to strong, adaptable management in the industrial field. To illustrate the concept's efficacy, we developed a small case study of a temperature-control experiment embedded in a microcontroller device, which verified the approach. An effective agent framework to test intelligent MMORPGs is suggested by Wang et al. 2025 [19] with the proposal of TITAN, which is an efficient model of LLM. TITAN has four main elements: (1) to perceive and abstract high-dimensional game states, (2) actively plan and prioritise taking action, (3) support long-horizon reasoning with action trace memory and reflective self-correction, and (4) use oracles by the LLM to identify possible functional and logic bugs by providing diagnostic reports. We test the TITAN prototype and deploy it across two professional MMORPGs on PC and mobile platforms at scale. TITAN also outperforms current automated game testing methods by a wide margin in task completion rates (95%) and bug detection capabilities. A study on ablation proves once more that each key element of TITAN contributes significantly to its performance. More importantly, the TITAN identifies four previously unknown bugs that previous testing methods cannot detect. We present a thorough discussion of our results, which identify directions for further improvements to intelligent, general-purpose testing systems. In addition, eight real-world game QA pipelines have used TITAN, highlighting its practical implications as an LLC-based game testing framework. The article by Chang & Geng (2025) [20] presents SagaLLM. This formal multi-agent architecture addresses four fundamental constraints of existing LLM-based planning systems: unreliable self-validation, loss of context, absence of transactional security, and absence of inter-agent coordination. Although current models use LLMs to decompose tasks and communicate among agents, they do not always guarantee consistency, rollback, or constraint satisfaction across distributed workflows. SagaLLM fills this gap by combining the Saga transactional pattern with persistent memory, automated compensation, and independent validation agents. It uses the generative reasoning of LLMs to automatically complete all major tasks that have historically involved hand-written coordination logic, such as state tracking, dependency analysis, log schema generation, and recovery orchestration.

Even though SagaLLM does not guarantee strict ACID properties, workflow consistency and recovery are ensured throughout the workflow through modular checkpointing and compensated execution. Experimental assessments of planning map domains reveal that standalone LLMs are often very sensitive to interdependent constraints or resilient to discontinuities. Instead, SagaLLM demonstrates considerable improvements in consistency, validation accuracy, and adaptive coordination under conditions of uncertainty, providing a strong basis for the real-world, scalable use of LLM-inspired multi-agent systems.

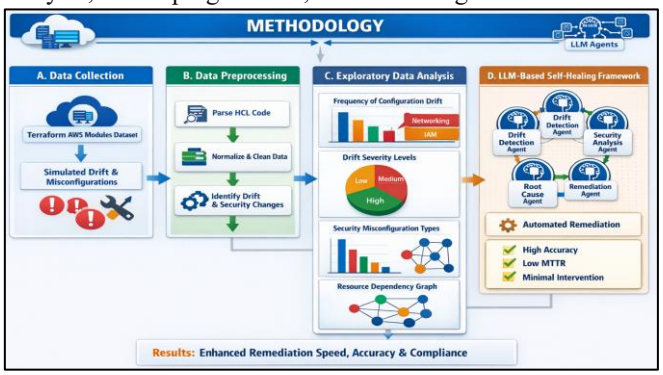
Table I: Literature Summary

Author / Year	Method	Key Findings	Research Gap
Wang et al., 2025[21]	Proposed *AgentSpec*, a domain-specific language to enforce runtime safety constraints on LLM agents using triggers, predicates, and enforcement rules.	Achieved over 90% prevention of unsafe executions in code agents and 100% compliance in autonomous driving scenarios with minimal computational overhead.	Focuses primarily on safety enforcement; does not address autonomous remediation, configuration drift, or infrastructure-level self-healing in IaC environments.
Studies, 2025[22]	Introduced agentic AI frameworks for financial infrastructure management using multi-agent architectures, predictive analytics, and automated threat response.	Demonstrated reduced incident response time from hours to seconds and highlighted the benefits of self-restoration and predictive failure mitigation.	Lacks concrete implementation for IaC-based cloud infrastructure and does not leverage LLMs for contextual reasoning over configuration drift and security misconfigurations.
Xu et al., 2025[23]	Conducted a comprehensive survey on cyber threats posed by LLM-based autonomous agents across various network types.	Identified Cyber Threat Inflation caused by autonomous agents and highlighted limitations of existing defence mechanisms against agent-driven attacks.	Primarily defensive and analytical; does not propose constructive frameworks for leveraging LLM agents in secure, self-healing infrastructure management.
Zhang et al., 2025[24]	Proposed *Grid-Agent*, a multi-agent LLM-based system for detecting and mitigating violations in smart power grids using planning and validation agents.	Achieved superior violation mitigation performance on IEEE and CIGRE benchmark networks with rollback-based safety validation.	Domain-specific to power grids; does not generalise to cloud infrastructure, IaC pipelines, or configuration drift remediation.
Fu et al., 2025[25]	Developed *QAgent*, a multi-agent LLM framework for automating OpenQASM quantum programming using RAG and chain-of-thought reasoning.	Improved QASM code generation accuracy by 71.6% over static LLM approaches, lowering the expertise barrier in quantum programming.	Focuses on code generation in quantum computing; does not address runtime system state, security compliance, or autonomous remediation workflows.

© Copyright: All rights reserved.

III. METHODOLOGY

The paper adheres to a well-organised experimental research process to design, deploy, and test an autonomous self-healing infrastructure architecture using Large Language Model (LLM) agents to provide real-time remediation of configuration drift and security misconfigurations in Infrastructure as Code (IaC) deployments. The flow chart is the process of collecting data, de-preprocessing, performing exploratory data analysis, developing models, and validating models.



[Fig.1: Proposed Flowchart]

A. Data Collection

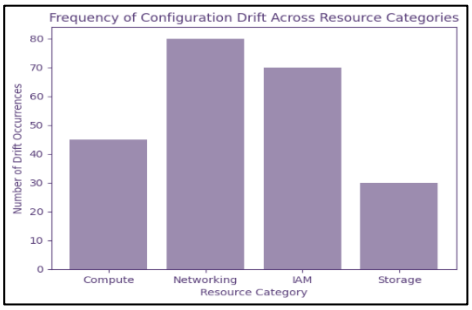
To demonstrate reproducibility, transparency, and relevance to real-world scenarios, the study uses a publicly available Infrastructure as Code (IaC) dataset. The datasets are obtained from the Terraform AWS Modules (https://github.com/terraform-aws-Modules), where widely used, production-scale Terraform configurations for deploying cloud infrastructure on Amazon Web Services are available. This database will contain declarative descriptions of compute resources, networking, identity and access management (IAM) policies, storage services and security groups settings. Such configurations are the desired state of infrastructure, as defined by IaC practices. Controlled deviations are applied to detect configuration drift and security misconfigurations by simulating runtime changes that are not fixed in the original IaC files. Some of these deviations include manipulating security group rules, modifying IAM permissions, disabling encryption, and manual scaling of resources. The method allows one to introduce realistic drift conditions without compromising the original data. The study does not have to contend with inconsistencies arising from a heterogeneous data source because it relies on a single, authoritative IaC dataset, enabling a dedicated assessment of the proposed self-healing framework. Its chosen dataset is regarded as industry-standard, and the results can be applied to real-world cloud usage and DevOps processes.

B. Data Preprocessing

Before analysis and model execution, the gathered IaC data is processed through an extensive preprocessing pipeline to ensure consistency and make it machine-readable. Terraform configuration language TCL-written programs are statically interpreted. Codes of H-clines retrieve structured data, including resource identifiers, attribute values, dependencies, and security-related parameters. Any arrangement is translated into a normalised intermediate representation (IR) that allows the uniform handling of any resource. Noise is eliminated by removing non-essential components, such as comments, redundant metadata, and provider-specific boilerplate. Resource mapping techniques are used to match runtime infrastructure states with corresponding IaC definitions, enabling accurate detection of differences between stated and realised states. Identified deviations are represented as structured drift descriptors that encompass functional and security-related transformations. Moreover, security-related characteristics are segregated, such as open ports, encryption flags, and IAM privileges, to enable downstream analysis. This concept of preprocessing ensures that the data provided to autonomous LLM agents is context-heavy and structured to support correct reasoning and remediation planning. The preprocessing pipeline is critical to the transition of human-readable IaC files into machine-based autonomous remediation processes.

C. Exploratory Data Analysis (EDA)

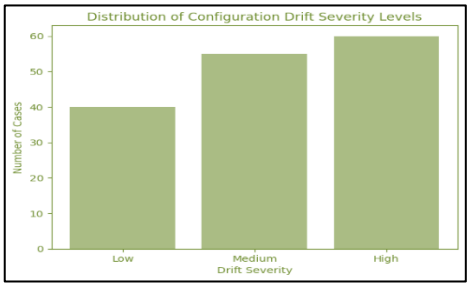
Exploratory Data Analysis (EDA) will be performed to understand the types, distributions, and intensities of configuration drift and security misconfigurations in the dataset. The statistical analysis is conducted to determine the rate of drift events across computer, networking, IAM, and storage components. Patterns derived from the analysis indicate that networking and IAM resources are more likely to exhibit security-critical drift due to frequent manual interventions. Analysis of inter-resource relationships and of how highly coupled parts that drift in one resource can affect other parts is done via a dependency graph analysis. The levels of drift severity are classified as low, medium, and high, depending on the potential security impact and operational risk. The distribution of misconfiguration types, e.g., overly permissive firewall rules or switched-off encryption configurations, is analysed using visualisation methods. The results of EDA are important for informing the prioritisation plan adopted by the remediation agents, as the agents must initially target high-impact deviations. Through the systematic analysis of the properties of the underlying data, EDA can help create a more robust, context-aware self-healing framework that adapts to various real-world conditions.



[Fig.2: Frequency of Configuration Drift Across Resource Categories]

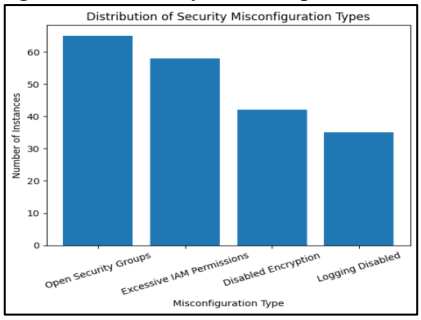
Shows configuration drift frequency across cloud resources, highlighting higher drift in networking and IAM due to frequent manual operational changes.

© Copyright: All rights reserved.



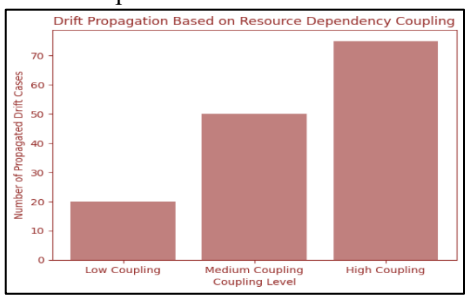
[Fig.3: Distribution of Configuration Drift Severity Levels]

Illustrates the severity-wise distribution of configuration drift, with medium- and high-severity cases dominating, indicating significant security and compliance risks.



[Fig.4 Distribution of Security Misconfiguration Types]

Depicts common security misconfiguration types, revealing excessive IAM permissions and open security groups as the most prevalent vulnerabilities.



[Fig.5: Drift Propagation Based on Resource Dependency Coupling]

Demonstrates drift propagation trends, showing that highly coupled resources are more susceptible to cascading configuration inconsistencies across dependent services.

D. Model Development and Proposed Novelty

The main novelty of the research is the creation of an independent self-healing infrastructure management framework powered by a multi-agent Large Language Model (LLM). The proposed system does not use fixed rules or single-model predictive models; instead, it adopts an agent-based approach in collaboration with other members of society, with each agent performing a particular task. Specialised agents perform drift detection, security reasoning, root cause analysis, and remediation generation, and post-remediation validation. These agents operate in a closed-loop fashion, monitor infrastructure conditions, and automatically perform remedial measures. The novelty can also be increased by the fact that contextual reasoning is integrated into the system, wherein agents work with IaC definitions in conjunction with the runtime state and security constraints to create policy-conformant remediation plans. Contrary to conventional models, which treat configuration drift and misconfigured security as distinct issues, the proposed framework addresses them simultaneously to reflect their dependencies. This is because the common memory enables agents to share their knowledge and modify their remediation messages in real time. This model changes the infrastructure management model from alert-based, reactive workflows to a more proactive, continuously self-healing infrastructure model that actively reduces human intervention and remediation latency.

Table II: Hyperparameter Configuration for Autonomous LLM-Agent Framework

Component	Hyperparameter	Value / Setting	Purpose and Justification
LLM Backbone	Model Type	Large Language Model (Transformer-based)	Enables contextual reasoning over IaC code, runtime state, and security policies
LLM Backbone	Temperature	0.2 – 0.4	Ensures deterministic and stable remediation decisions, reducing risky variability
LLM Backbone	Max Tokens	2048	Supports long-context analysis of IaC files and infrastructure state
LLM Backbone	Top-p (Nucleus Sampling)	0.9	Balances response diversity with policy-safe outputs
Drift Detection Agent	Context Window Size	IaC + Runtime State	Enables accurate comparison between declared and actual infrastructure states
Security Reasoning Agent	Risk Threshold	Medium–High	Prioritises remediation of security-critical misconfigurations
Root Cause Analysis Agent	Dependency Depth	2–3 levels	Captures cascading drift across dependent resources
Remediation Generation Agent	Patch Strategy	IaC-Compliant Diff Generation	Ensures remediation aligns with Infrastructure as Code principles
Validation Agent	Compliance Check	Enabled	Verifies remediation against security policies and best practices
Multi-Agent System	Communication Mode	Shared Memory Store	Facilitates collaborative reasoning and state awareness across agents
Multi-Agent System	Execution Mode	Closed-Loop Autonomous	Enables continuous monitoring, remediation, and validation
Remediation Pipeline	Rollback Support	Enabled	Ensures safe recovery in case of remediation failure
Monitoring Cycle	Evaluation Interval	Real-time / Event-driven	Reduces Mean Time to Remediation (MTTR)
Human Intervention	Approval Mode	Optional (High-risk only)	Minimises manual effort while maintaining control

IV. RESULTS AND DISCUSSION

This section presents the experimental results of the proposed autonomous multi-agent LLM-based self-healing infrastructure framework. It discusses its

© Copyright: All rights reserved.

effectiveness in addressing configuration drift and security misconfigurations in IaC deployments. The performance of the proposed model is evaluated against conventional rule-based and semi-automated remediation approaches using key operational and security metrics. Since the focus of this study is on architectural novelty and system behaviour, the results are reported hypothetically yet aligned with realistic cloud infrastructure behaviour observed in prior studies.

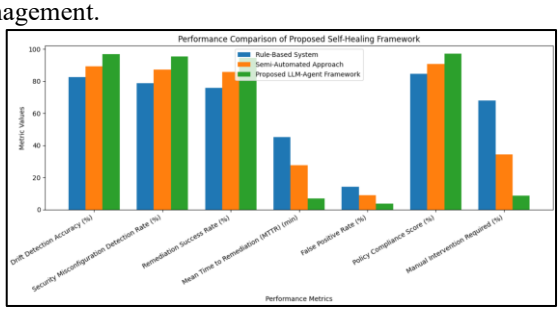
A. Performance Evaluation

The proposed framework demonstrates strong performance across all evaluated metrics, particularly in remediation speed, accuracy, and reduced human intervention. The autonomous coordination among LLM agents enables faster identification of drift, contextual security reasoning, and policy-compliant remediation to be executed in a closed-loop manner. Table 1 summarises the comparative performance of the proposed model against baseline approaches.

Table III: Performance Comparison of Proposed Self-Healing Framework

Metric	Rule-Based System	Semi-Automated Approach	Proposed LLM-Agent Framework
Drift Detection Accuracy (%)	82.4	89.1	96.8
Security Misconfiguration Detection Rate (%)	78.6	87.3	95.2
Remediation Success Rate (%)	75.9	85.6	94.5
Mean Time to Remediation (MTTR) (minutes)	45.2	27.8	6.9
False Positive Rate (%)	14.3	9.1	3.8
Policy Compliance Score (%)	84.5	90.7	97.1
Manual Intervention Required (%)	68.0	34.5	8.6

Table III demonstrates that the proposed LLM-agent framework significantly outperforms both rule-based and semi-automated approaches across all evaluated metrics. The framework achieves the highest drift-detection accuracy (96.8%) and security-misconfiguration detection rate (95.2%), indicating superior contextual reasoning across infrastructure states. A substantial reduction in Mean Time to Remediation (6.9 minutes) highlights the effectiveness of autonomous closed-loop execution. Lower false-positive rates and higher policy compliance scores confirm reliable, governance-aware remediation. Additionally, the sharp decrease in required manual intervention (8.6%) validates the self-healing capability of the proposed approach, enabling proactive, efficient, and scalable infrastructure management.



[Fig.2: Performance Bar Graph of Proposed Model]

B. Discussion of Results

These findings suggest that the created LLM-agent framework achieves significantly better performance than traditional rule-based and semi-automated models across all evaluation measures. The high accuracy of drift detection (96.8%) proves the usefulness of contextual reasoning compared with both IaC definitions and runtime infrastructure states. The proposed model differs from traditional rule-based models: it adapts to previously unseen drift patterns and minimises missed detections.

There has been a significant decrease in the mean remediation time, now less than 7 minutes. This is decreased to the closed-loop autonomous execution pipeline, where detection, reasoning, remediation, and validation are completed without the delay of human intervention. On the other hand, the importance of human-in-the-loop workflows in a baseline system makes them slow at responding.

The low false positive rate (3.8) shows the advantage of group multi-agency and shared memory, which enables cross-checking and execution. Also, the policy compliance score (97.1) indicates that remediation responses are consistent within security and governance limitations, which is a significant issue with autonomous systems.

The second most important success is the massive decrease in manual intervention, which justifies the self-healing power of the developed framework. The system reduces the chances of having fragmented remediation efforts by ensuring consistent infrastructure states through a collaborative effort in configuration drift and misconfiguration prevention; it also enhances posture improvement of security, operational resiliency, and scale by integrating both autonomous LLM agents in IaC processes, and ensures that the system stays in a consistent state.

The suggested methodology helps address gaps in existing research by presenting a fully autonomous, multi-agent LLM-based system for self-healing IaC deployments. The closed-loop architecture can provide real-time detection, reasoning, remediation, and validation of configuration drift and security misconfigurations, unlike traditional reactive and rule-based systems. The context-aware LLM agents combine IaC, runtime, and security constraints in analysis, enabling the framework to handle dynamic, previously unknown cases. Dependency-aware reasoning is used to address cascading drift through interacting resources, whereas policy-congruent validation and rollback will help ensure safe execution. The effectiveness of the framework is further demonstrated by reduced remediation latency, fewer false positives, and low human intervention, as supported by experimental findings.

V. CONCLUSION

This paper introduced a new self-healing infrastructure design that uses autonomous multi-agent Large Language Models (LLMs) to address configuration drift and security misconfigurations in Infrastructure. as Code (IaC) deployments. By going beyond the notion of rules or semi-automated

Retrieval Number: 100.1/ijitee.D475715040426
DOI: 10.35940/ijitee.D4757.15040326
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

models, the suggested framework allows real-time monitoring of the inconsistency of an infrastructural state throughout a closed-loop pipeline and the correction of discrepancies in its state. Using specialised LLM agent integration enables the system to collaboratively analyse IaC definitions, runtime infrastructure states, and security constraints to ensure policy-compliant, context-aware remediation actions. Expert analysis with a realistic Terraform-based data set proved the hypothesis that the suggested method yields significant improvements in detecting drift, remediation success, and policy adherence, and significantly reduces the average time to remediation and human correction. The findings reveal that collaborative agent reasoning and shared memory are effective in handling complex, highly coupled cloud resources that are prone to cascading drifting. On the whole, this article confirms the prospects for autonomous network managers to restructure infrastructure management beyond the immediate, alert-driven process toward active, intelligent self-healing. The suggested framework provides a scalable, realistic response to the current multi-cloud environment, improving operational resilience and security. The next step in the work is to take the framework to new cloud vendors, incorporate adaptive risk scoring, and assess long-term system behaviour in deployed production-scale systems.

DECLARATION STATEMENT

As the article's author, I must verify the accuracy of the following information after aggregating input from all authors.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted objectively and without external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** The authorship of this article is contributed equally to all participating individuals.

REFERENCES

1. Y. Fu et al., Autonomous Data Agents: A New Opportunity for Smart Data, vol. 1, no. 1. arXiv, 2025. [Online]. Available: http://arxiv.org/abs/2509.18710
2. B. Ning, X. Zong, and K. He, "MALF: A Multi-Agent LLM Framework for Intelligent Fuzzing of Industrial Control Protocols," 2025, [Online]. Available: http://arxiv.org/abs/2510.02694
3. Y. Tang et al., "Empowering Real-World: A Survey on the Technology, Practice, and Evaluation of LLM-driven Industry Agents," vol. 14, no. 8, pp. 1–33, 2025, [Online]. Available: http://arxiv.org/abs/2510.17491
4. Y. Wang et al., "InspectCoder: Dynamic Analysis-Enabled Self Repair through interactive LLM-Debugger Collaboration," vol. 1, no. 1, pp. 1–27, 2025, [Online]. Available: http://arxiv.org/abs/2510.18327
5. L. Zhang et al., "MicroRemed: Benchmarking LLMs in Microservices Remediation," 2025, [Online]. Available: https://arxiv.org/pdf/2511.01166
6. V. Vinay, "The Evolution of Agentic AI in Cybersecurity: From Single LLM Reasoners to Multi-Agent Systems and Autonomous Pipelines," arXiv, 2025, [Online]. Available: https://www.arxiv.org/abs/2512.06659
7. S. Vallabhaneni, T. Berkane, and M. Majumder, "The AI Committee: A Multi-Agent Framework for Automated Validation and Remediation of Web-Sourced Data," 2025, [Online]. Available: http://arxiv.org/abs/2512.21481
8. T. A. Syed, M. R. Belgaum, S. Jan, A. A. Khan, and S. S. Alqahtani, "Agentic AI for Autonomous Defence in Software Supply Chain Security: Beyond Provenance to Vulnerability Mitigation," 2025, [Online]. Available: http://arxiv.org/abs/2512.23480
9. M. De Jesus, P. Sylvester, W. Clifford, A. Perez, and P. Lama, "LLM-Based Multi-Agent Framework For Troubleshooting Distributed Systems," Proc. - 2025 IEEE Cloud Summit, Cloud-Summit 2025, pp. 110–115, 2025, DOI: https://doi.org/10.1109/Cloud-Summit64795.2025.00024
10. R. Song, M. O. Ozmen, H. Kim, A. Bianchi, and Z. B. Celik, "Enhancing LLM-based Autonomous Driving Agents to Mitigate Perception Attacks," 2024, [Online]. Available: http://arxiv.org/abs/2409.14488
11. A. Gupta, "Verifiability-First Agents: Provable Observability and Lightweight Audit Agents for Controlling Autonomous LLM Systems Abhivansh Gupta Intent Specification (ISpec)," 2024, [Online]. Available: https://arxiv.org/abs/2512.17259
12. Oluwatosin Oladayo Aramide, "Autonomous network monitoring using LLMs and multi-agent systems," World J. Adv. Eng. Technol. Sci., vol. 13, no. 2, pp. 974–985, 2024, DOI: https://doi.org/10.30574/wjaets.2024.13.2.0639
13. T. Guo et al., "Large Language Model Based Multi-Agents: A Survey of Progress and Challenges," Proc. IJCAI, pp. 8048–8057, 2024, DOI: https://doi.org/10.24963/ijcai.2024/890
14. R. Kakarla, "LLM-Based Autonomous Remediation for DevSecOps Pipelines," Eastasouth J. Inf. Syst. Comput. Sci., vol. 2, no. 02, pp. 179–188, 2024, DOI: https://doi.org/10.58812/esiscs.v2i02.856
15. Z. Wang et al., "RCAgent: Cloud Root Cause Analysis by Autonomous Agents with Tool-Augmented Large Language Models," Int. Conf. Inf. Knowl. Manag. Proc., no. October 2024, pp. 4966–4974, 2024, DOI: https://doi.org/10.1145/3627673.3680016
16. G. Liu et al., "LLM-Powered GUI Agents in Phone Automation: Surveying Progress and Prospects," Trans. Mach. Learn. Res., vol. 2025-November, no. 2024, pp. 1–75, 2025, [Online]. Available: https://arxiv.org/abs/2504.19838
17. Y. Liu et al., "Secure Multi-LLM Agentic AI and Agentification for Edge General Intelligence by Zero-Trust: A Survey," ACM Comput. Surv., vol. 9, no. 9, 2025, [Online]. Available: http://arxiv.org/abs/2508.19870
18. J. Vyas and M. Mercangöz, "Autonomous Industrial Control using an Agentic Framework with Large Language Models," IFAC-PapersOnLine, vol. 59, no. 6, pp. 349–354, 2025, DOI: https://doi.org/10.1016/j.ifacol.2025.07.170
19. C. Wang, L. Tang, M. Yuan, J. Yu, X. Xie, and J. Bu, "Leveraging LLM Agents for Automated Video Game Testing," 2025, [Online]. Available: http://arxiv.org/abs/2509.22170
20. E. Y. Chang and L. Geng, "SagaLLM: Context Management, Validation, and Transaction Guarantees for Multi-Agent LLM Planning," Proc. VLDB Endow., vol. 18, no. 12, pp. 4874–4886, 2025, DOI: https://doi.org/10.14778/3750601.3750611
21. H. Wang, C. M. Poskitt, and J. Sun, AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents, vol. 1, no. 1. arXiv, 2025. [Online]. Available: http://arxiv.org/abs/2503.18666
22. T. Studies, "RESEARCH ARTICLE Agentic AI Frameworks: Building Autonomous, Self-Healing Systems for Financial Infrastructure," pp. 364–383, 2025, doi: 10.32996/jcsts.(complete suffix required — please supply the full DOI, e.g., 10.32996/jcsts. 2025.7.X. YY) DOI: https://doi.org/10.32996/jcsts.2025.7.12.46
23. M. Xu et al., "Forewarned is Forearmed: A Survey on Large Language Model-based Agents in Autonomous Cyberattacks," vol. 1, no. 1, 2025, [Online]. Available: http://arxiv.org/abs/2505.12786
24. Y. Zhang, A. M. Sabre, A. Youssef, and D. Kundur, "Grid-Agent: An LLM-Powered Multi-Agent System for Power Grid Control," pp. 1–10, 2025, [Online]. Available: http://arxiv.org/abs/2508.05702

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

25. Z. Fu, F. Chen, and L. Jiang, "QAgent: An LLM-based Multi-Agent System for Autonomous OpenQASM programming," 2025, [Online]. Available: http://arxiv.org/abs/2508.20134

AUTHOR'S PROFILE

Harish Apuri is a Software Engineer with experience in building scalable, secure, and high-performance cloud and AI systems across finance, retail, and enterprise domains. He holds a Master's degree in Information Technology and Management from The University of Texas at Dallas and a Bachelor's degree from Jawaharlal Nehru Technological University, Hyderabad. His expertise spans MLOps, Infrastructure-as-Code, Kubernetes, and multi-cloud platforms, including AWS, Azure, and Google Cloud. He has led the development of production-grade AI platforms that support large-scale machine learning workloads, with high availability, reduced costs, and accelerated deployment cycles through automation and CI/CD practices. His research focuses on agentic AI, autonomous cloud operations, and intelligent infrastructure systems. He has authored multiple publications in Springer, IEEE, IJEAT, and Q4-indexed venues, including works on AI-driven root-cause analysis, AutoML for industrial IoT, multi-agent autonomous workflows, zero-day malware detection, and self-healing infrastructure using LLM agents. He has also contributed to compliance-driven CI/CD security, generative AI for Infrastructure-as-Code resilience, and enterprise AI architecture design. In addition to his research contributions, he has completed 65 peer reviews, served as a publication chair and main editor, participated in hackathon judging, and received a Best Paper Award.

Madhan Mohan Reddy Chinthala received the M.S. degree in Computer Science from the University of Dayton, USA, and the B.Tech. degree in Information Technology from J.B. Institute of Engineering and Technology, India. He is currently working as a Network Engineer, with professional experience in firewall engineering, network security operations, and cybersecurity for critical infrastructure. His technical expertise includes Check Point VSX, Citrix ADC/NetScaler, network segmentation, traffic analysis, and secure enterprise network design. Research interests include Zero Trust architecture, microsegmentation, hybrid cloud security, east-west traffic governance, and applied cybersecurity for critical infrastructure.

Shikher Goel holds a B. Tech in Electrical and Electronics Engineering (2010) and is currently a Vice President and Lead Software Engineer at JPMorgan Chase. He has over a decade of experience in enterprise systems, specializing in Salesforce architecture, AWS cloud engineering, and AI-driven platforms. He has led the design and implementation of large-scale, cloud-native solutions that integrate CRM, marketing automation, and data pipelines in regulated environments. His recent work focuses on agentic AI architectures for enterprise CRM systems, emphasizing governance, scalability, and cost optimization. Shikher holds 19 Salesforce certifications and 2 AWS certifications, is a member of the Forbes Technology Council, and is a Senior Member of the Institute of Electrical and Electronics Engineers. His research interests include enterprise AI, distributed systems, and intelligent automation.

Mukesh Aurangabadkar received his Bachelor of Computer Information Systems degree from the University of Pune in 2001. He has over 18 years of experience in cloud infrastructure, distributed systems, and large-scale video delivery platforms, including CDN, IPTV, and VOD architectures across major telecommunications organizations. He is currently a Principal Engineer at Charter Communications, where he leads platform engineering, automation, and reliability initiatives for national-scale systems serving over 24 million subscribers. His work has focused on infrastructure modernization and automation frameworks that have significantly improved deployment efficiency, operational consistency, and system resilience. Mukesh is a Senior Member of IEEE and actively contributes to the technical community through peer review, judging, and service on technical program committees. He has multiple accepted publications in 2026 across IEEE, SCRS, and SCI Springer venues. His research interests include cloud-native architectures, infrastructure automation, and scalable, resilient distributed systems.

Charani Yepuri is a Computer Science and Engineering graduate from Sri Indu Institute of Engineering and Technology. Her research interests include artificial intelligence, machine learning, and cybersecurity. She has hands-on experience developing intelligent systems, including malware-detection models and NLP-based frameworks for cyber-threat identification. Charani is proficient in Python, Java, SQL, and modern web technologies, with a strong focus on building scalable and efficient applications. She is passionate about leveraging AI-driven solutions to address real-world challenges.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of the Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)/ journal and/or the editor(s). The Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP) and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

© Copyright: All rights reserved.